Artificial learning approaches for multi-target tracking

Douglas Blount[a], Michael Kouritzin[b], and Jesse McCrosky[b]

[a]Department of Mathematics at Arizona State University, Temps, USA;
[b]Prediction in Interacting Systems, Mathematics of Information Technology and Complex Systems, Department of Mathematics at the University of Alberta, Edmonton, Canada

ABSTRACT

A hybrid weighted/interacting particle filter, the selectively resampling particle (SERP) filter, is used to detect and track an unknown number of independent targets on a one-dimensional "racetrack" domain. The targets evolve in a nonlinear manner. The observations model a sensor positioned above the racetrack. The observation data takes the form of a discretized image of the racetrack, in which each discrete segment has a value depending both upon the presence or absence of targets in the corresponding portion of the domain, and upon lognormal noise. The SERP filter provides a conditional distribution approximated by particle simulations. After each observation is processed, the SERP filter selectively resamples its particles in a pairwise fashion, based on their relative likelihood. We consider a reinforcement learning approach to control this resampling. We compare two different ways of applying the filter to the problem: the signal measure approach and the model selection approach. We present quantitative results of the ability of the filter to detect and track the targets, for each of the techniques. Comparisons are made between the signal measure and model selection approaches, and between the dynamic and static resampling control techniques.

Keywords: Particle filter, Nonlinear filtering, SERP filter, Stochastic processes, Neural networks, Artificial learning, Racecar model, Model selection

1. INTRODUCTION

Filtering theory is an active research field with a wide range of applications, including target detection and tracking, asset pricing and portfolio allocation, signal processing, pollution tracking, and search and rescue. Filtering theory allows one to find a probabilistic distribution of the past, present, and future state of a signal based on observations that may be partial, corrupted and/or noisy.

For some simple applications there exist optimal analytic solutions; however, most real-world applications involve nonlinear signals for which optimal solutions cannot readily be computed. For problems such as these, approximate solutions, such as particle filters, are used. Particle filters use Monte Carlo simulations to approximate the true filtering equation. The estimate of particle filters approaches the optimal filter as the number of particles used approaches infinity.

In this paper we consider a multiple target problem that we refer to as the racecar problem. It involves detecting and tracking an unknown number of independent targets. There are various ways of applying a particle filter to a multi-target problem. We explore two techniques: the signal measure approach, which is the canonical choice, and the model selection approach. We also consider the use of an artificial neural network to improve the performance of our particle filter.

This paper is organized as follows: Section 2 describes the filtering methods we use. In Section 3, the problem is presented as a signal and observation model. Section 4 describes the application of our filtering techniques to the problem and provides simulation data that compares the filtering performance in terms of error measures and computation time. Lastly, in Section 5, conclusions are presented.

Further author information: (Send correspondence to M.K.)
D.B.: E-mail: blount@math.la.asu.edu, Telephone: 1 602 965 2641
M.K.: E-mail: mkouritz@math.ualberta.ca, Telephone: 1 780 492 2704
J.M.: E-mail: mccrosky@math.ualberta.ca, Telephone: 1 780 492 0215

Automatic Target Recognition XIV, edited by Firooz A. Sadjadi, Proceedings of SPIE Vol. 5426
(SPIE, Bellingham, WA, 2004) · 0277-786X/04/$15 · doi: 10.1117/12.542674

2. FILTERING METHODS

2.1. Background

The goal of our filtering methods is to determine the conditional distribution

$$P(X_{t_k} \in A | Y_1, \ldots, Y_k) \tag{1}$$

of X_{t_k}, which is the signal state at time t_k, based on the back-observations, $Y_1, \ldots, Y_k$, where Y_i is the observation data from time step i. In our case, X is a multi-target signal. There are many ways to use a particle filter to find this conditional distribution. Herein, two methods are considered. The signal measure approach uses a counting measure on the number of targets in the domain. The model selection approach runs a separate instance of the filter for each possible number of targets and uses Bayesian model selection to weight and prune which models are considered.

2.2. SERP Filter

The SERP filter is a hybrid weighted/interacting particle filter, first discussed in Ballantyne *et al*,[4] then known as the weighted interacting or hybrid particle filter. The filter's applicability to performing arts and to search and rescue was demonstrated in Bauer *et al*,[5] and in Ballantyne *et al*.[6]

During the resampling step, the filter uses a parameter, ρ, to control how much resampling is performed. This is the parameter that we attempt to optimize in section 2.5.

2.3. Signal measure approach

For the signal measure approach, the signal, X, is considered a counting measure of the number of targets, M^X, in the domain,

$$X = \sum_{j=1}^{M^X} \delta_{X^j} \tag{2}$$

where δ is the Dirac delta measure.

To implement this approach with the SERP filter, each particle is initialized with a number of targets sampled from the initial signal distribution. Each of the targets belonging to each particle is evolved independently. The particle distribution then covers the entire distribution of the signal, including each possible number of targets.

This technique reduces the problem to a single-target problem with an expanded state space, and allows normal particle filter algorithms to be applied.

2.4. Model selection approach

To use the SERP filter for model selection, an instance of the filter is created for each possible number of targets. Typically, each filter is normalized to produce a proper probability measure of the signal state:

$$P(X_t \in A) = \sum_{i=1}^{N} \frac{W_t^i \times 1_{X_t^i \in A}}{\sum_{i=1}^{N} W_t^i}, \tag{3}$$

where N is the number of particles, W_t^i is the weight of particle i and X_t^i is the state of particle i.

The denominator in Equation (3) can be considered a total weight of the unnormalized filter. This total weight can be used to calculate a Bayes' factor comparing the likelyhood of two models. The Bayes factor comparing filter i to filter j can be calculated as:

$$B_{i,j} = \frac{\sum_{k=1}^{N} W_t^{i,k}}{\sum_{k=1}^{N} W_t^{j,k}}, \tag{4}$$

where $W_t^{i,k}$ is the weight of particle k of filter i.

In actual implementation, instead of using the Bayes factor directly, the total weight of each filter is calculated along with a total weight for all active filters. A probability measure on the filters is then defined as:

$$P(model\ i\ is\ correct) = \frac{W_i}{W_{ttl}}, \tag{5}$$

where W_i is the total weight of filter i and W_{ttl} is the total weight of all models.

When the probability for a model falls below a certain threshold, it is deactivated, and its particles are assigned to the remaining models in proportion relative to their current weights. This last step keeps the computational time fairly constant.

This implementation is functionally equivalent to the Bayes factor method.

2.5. Rho Optimization

As described in section 2.2, the SERP filter makes use of a parameter ρ to control the amount of resampling performed. The optimal value for this parameter varies depending on the current state of the filter. For example, when the filter is first initialized, it needs to explore the signal space, and thus little resampling is appropriate; however, when the signal state has been localized, large amounts of resampling should be used to refine the estimate and ensure that the target is not lost.

Given this situation, we wish to determine a function that maps filter state to optimal ρ value.

2.5.1. State variables

The state of the filter at time step t_k is denoted by a set of 4 state variables $S_{t_k} = \{s_{t_k}^1, \ldots, s_{t_k}^4\}$ which attempt to capture information about the state of the particle system at time t_k. The first is an overall variance of the particle system, denoted *var*. Second, we use the variance of the filter distribution for the number of targets present, denoted var^t. Third, we use the change in the median weight of the filter between the last two iteration, denoted $\delta median$, and lastly we use the N-effective value, as discussed in Liu and Chen,[8] denoted N_{eff} and defined as:

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{n} \left(\frac{W^i}{\sum_{j=1}^{n} W^j} \right)^2}, \tag{6}$$

where W^i is the weight of particle i.

State variables are sampled in discrete intervals. These sample intervals are geometric for each state variable, except for $\delta median$, which uses arithmetic intervals.

2.5.2. Rho

We define a discretization of ρ_{t_k} by $\rho_{t_k} \in \{\rho_{t_k}^i : 0 \leq i < Q\}$, where Q is the number of discrete values that ρ_{t_k} can take. These values are chosen geometrically.

2.5.3. Policies and optimal policies

A policy π is a function that maps states S_{t_k} to actions ρ_{t_k}. The simplest policy is $\pi_s(S_{t_k}) = \varrho$, where ϱ is a constant value. At a given time t_k, the filter is in state S_{t_k}. A policy π is used to determine which ρ_{t_k} to use as the resampling parameter. Using $\rho_{t_k} = \pi(S_{t_k})$ for resampling, a filter cycle (resample, evolve, and reweight) is applied, leaving the filter in some new state $S_{t_{k+1}}$. We define the cost function for an action as:

$$C(\rho_{t_k}) = \varepsilon(F_{t_{k+1}}), \tag{7}$$

where $F_{t_{k+1}}$ is the conditional distribution of the filter at time t_{k+1} and $\varepsilon(F_{t_{k+1}})$ is the mean-squared error (MSE) of the expected value. Other measures of cost such as computation time could also be incorporated.

The optimal greedy policy for time t_k is:

$$\pi_0^* \equiv \arg\min_\pi C(\pi(S_{t_k}))\ \forall S_{t_k}. \tag{8}$$

This policy will minimize the cost for the current iteration, but may put the filter in a state so that future iterations will perform poorly. To find a true optimal value, one would have to consider a time-horizon cost; however, in this paper, we consider only the greedy policy.

2.5.4. Methodology

In order to find the optimal policy, we first populate a data table that records the average cost of choosing each ρ value in each state. We do this by simulation, running the filter on synthetic data. We then choose the lowest cost ρ value for each state as our partial policy. This partial policy will be incomplete and noisy due to some states being encountered few or no times during the data collection. To complete the policy, we use this partial policy to train a neural network.

2.5.5. Data collection

For the data collection stage, we run the filter for a series of data collection epochs. Each epoch consists of some fixed number of filter iterations. For each epoch, a new signal is generated and the filter is reset. Each iteration, the filter state is recorded and a ρ value is chosen. After the iteration completes, the cost is calculated and stored in the table, indexed by the state experienced and the ρ used.

Once the data collection is completed, the lowest cost ρ value is extracted for each state.

2.5.6. Neural network

The extracted data is then used to train a neural network as described in Mitchell.[7]

The neural network used is a fully-connected, feed-forward, back-propagation network with one hidden layer. There is one input unit per state variable, which is four in our case. There are eight hidden units and a single output unit. The activation function is the hyperbolic tangent. The output is unthresholded. This type of setup is capable of approximating, to an arbitrary accuracy, any continuous, bounded function.

The state values and ρ values are all normalized before being used to train the neural network. The mean and standard deviation of the training data are stored, allowing the system to properly normalize the input and denormalize the output during actual use.

3. PROBLEM DESCRIPTION

Our problem is to detect the number of targets existing within our domain and track their state based on back-observations. The targets all evolve independently and according to the same model.

The problem is referred to as the racecar problem. It is based on an earlier model described in Yewchuk *et al.*[1]

3.1. Signal Model

The signal consists of M targets,

$$X_t = \{X_t^1, X_t^2, \ldots, X_t^M\}, \tag{9}$$

where M is uniformly distributed between 0 and 3. Each target consists of two state variables,

$$X_t^i = \{v_t^i, \theta_t^i\}. \tag{10}$$

Here, θ exits on a manifold,

$$\theta_t^i \in [0, 2\pi), \tag{11}$$

and will wrap around its domain from 0 to 2π and vice versa. The variable v is a velocity which takes values

$$v_t^i \in \{-S, S\}, \tag{12}$$

where S is a constant speed.

The variable θ has a uniform initial distribution and evolves according to the explicit solution

$$\theta_t^i = P(\int_0^t v_s^i ds + \sigma_s W_t^i), \tag{13}$$

where W_t^i is a standard Brownian motion, independent of the Brownian motions for other targets, σ_s is a constant, and P is a periodic function used to keep the target within the domain, defined by:

$$P(x) = x - 2\pi \left\lfloor \frac{x}{2\pi} \right\rfloor. \tag{14}$$

The velocity v has a uniform initial distribution and switches between its two possible values as a Poisson process:

$$v_t^i = S(2(N(t) \mathrm{mod}_2) - 1), \tag{15}$$

where N(t) is the number of events that have occurred at time t in a Poisson process with a constant rate, λ.

3.2. Observation Model

The observations are discrete time and take the form of a one-dimensional raster with n pixels:

$$Y_{t_k} = \{Y_k^0, Y_k^1, \ldots, Y_k^{n-1}\}. \tag{16}$$

Each pixel takes a value depending on the presence or absence of targets in the corresponding segment of the domain.

$$Y_{t_k}^i = ((I_1 \times 1_{\exists_j \theta_{t_k}^j \in C_i}) + (I_0 \times 1_{\forall_j \theta_{t_k}^j \notin C_i})) L_{t_k}^i, \tag{17}$$

where $L_{t_k}^i$ is a lognormal random variable with expectation $e^{\mu + \frac{\sigma_o^2}{2}}$ and standard deviation $\sqrt{e^{2\mu+\sigma_o^2}(e^{\sigma_o^2} - 1)}$, where μ and σ_o are constants and represent the mean and standard deviation of the underlying normal distribution. I_1 and I_0 are constants, and C_i is the subset of the domain covered by observation component i and is defined as:

$$C_i = [\frac{i2\pi}{n}, \frac{(i+1)2\pi}{n}), \tag{18}$$

where n is the number of pixels in the observation raster.

4. SIMULATIONS

4.1. Rho Optimization

The ρ optimization system requires extensive computing time to converge to a good approximation of the optimal policy. Due to limited hardware available, the ρ optimization system was only trained for 3000 epochs of 30 iterations each. This was probably not sufficient training to get a precise approximation of the optimal policy, but we believe it still produces a policy that demonstrates the major features of the optimal policy.

4.1.1. Optimal rho values

In order to produce the graphs of the optimal ρ values, the values are averaged over the other state elements. For example, the *var* graph values are averaged over all possible N_{eff}, var_t, and $\delta median$ values.

Figures 1 and 2 show the Optimal ρ values versus *var* value. There are two separate figures because different variance measures were used for the model selection filter and the signal measure filter. Figure 3 shows the optimal ρ values versus N_{eff} value. Figure 4 shows the optimal ρ values versus var^t value. Figure 5 shows the optimal ρ values versus $\delta median$ value.

Unsurprisingly, for the signal measure filter, higher *var* values result in higher ρ values. The high *var* value indicates that the filter does not have a good lock on the target and thus excessive resampling must be avoided. For the model selection filter, however, the *var* graph is quite different. We have not been able to explain this behaviour. It is possible it is due to poor training, or perhaps a software problem.

For the N_{eff} graph (Figure 3), it is likely that the minima in the graph are values that are experienced when the filter has stabilized its weights after locking onto the target. From these minima, the optimal ρ values increase more quickly as N_{eff} increases than as it decreases. It is likely that smaller N_{eff} are sometimes encountered by the filter when it has an especially stable lock.

The var^t graph (Figure 4) is also unsurprising. As with *var*, higher values indicate higher uncertainty of the signal state. In this case the model selection filter also performs as expected, unlike the *var* graphs.

The $\delta median$ graph (Figure 5) is difficult to analyze. The central peak suggest that perhaps when the particles are in diverse states, the total weight tends to decrease at a fairly constant rate, but then when a lock is achieved, the $\delta median$ varies more widely.

4.2. Method Comparisons

Each of our filtering methods was simulated in software using synthetic data. We use the computation time and an error measure, described below, to compare the filters.

4.2.1. Error measurement

Normally, the error of a particle filter approximation would be measured in terms of mean squared error (MSE) defined as:

$$\text{MSE}(t_k) = \frac{1}{r_{\max}} \sum_{r=1}^{r_{\max}} \text{d}(X_{t_k}^r, E[X_{t_k}^r | Y_1^r, \ldots, Y_k^r])^2, \tag{19}$$

where X^r, Y_k^r are the signal path and observations from run number r, $r_{\max}$ is the total number of simulation runs, and d is some distance function defined on the signal domain. However, in the case of multiple targets in which each particle X^j is a counting measure rather than a single point, and in which a distance between two counting measures or the mean of a set of counting measures which may contain differing number of points has no usual definition, no standard MSE calculation is possible and a different value for filter error must be defined.

Therefore, we define $\Upsilon_{t_k}(\left\{X_{t_k}^j\right\}_{j=1}^N, X_{t_k})$, the error of the filter measure with respect to signal truth, to be

$$\Upsilon_{t_k}(\left\{X_{t_k}^j\right\}_{j=1}^N, X_{t_k}) = \sum_{i=1}^N \frac{W_k^i}{\sum_{l=1}^N W_k^l} \text{d}(X_{t_k}, X_{t_k}^i), \tag{20}$$

where for $X = \frac{1}{M^X}\sum_{j=1}^{M^X}\delta_{X^j}$ and $Y = \frac{1}{M^Y}\sum_{j=1}^{M^Y}\delta_{Y^j}$,

$$\mathrm{d}(X,Y) = \begin{cases} \min_{\sigma\in\mathrm{perm}\{1,\ldots,M^Y\}} \sqrt{\sum_{j=1}^{M^X}\left(\Pi(X^j, Y^{\sigma(j)})\right)^2 + (M^Y - M^X)\,D^2} & M^X \le M^Y \\ \min_{\sigma\in\mathrm{perm}\{1,\ldots,M^X\}} \sqrt{\sum_{j=1}^{M^Y}\left(\Pi(X^{\sigma(j)}, Y^j)\right)^2 + (M^X - M^Y)\,D^2} & M^X > M^Y \end{cases}. \tag{21}$$

Here $\Pi(X^{j_1}, Y^{j_2})$ is the distance between the locations of the targets X^{j_1} and Y^{j_2}, and D is the diagonal length of the target domain with respect to this distance function Π.

In the case of the model selection filter we define an aggregate error measure over each of the component filters,

$$\Upsilon^{ms}_{t_k}(\left\{\mathbb{F}^j_{t_k}\right\}_{j=0}^{M}, X_{t_k}) = \sum_{i=1}^{M}\frac{W^j_k}{\sum_{j'=o}^{M} W^{j'}_k}\Upsilon_{t_k}(F^j_{t_k}, X_{t_k}), \tag{22}$$

where $F^j_{t_k}$ is the filter for the j-target case, M is the maximum possible number of targets, and W^j_k is the total weight of filter j.

4.3. Simulation Results

Four different filters were each simulated 200 times. Each simulation consisted of 200 runs of 60 observations with 0.5 seconds of simulation time between each observation. Each run, a number of targets for the signal was chosen uniformly between 0 and 3, according to the signal distribution. Each of the filtering techniques, signal measure and model selection, were run using both static and dynamic ρ selection. For each run, the computation time and error value for each iteration were measured. The values for each simulation were averaged over the runs. Figure 6 shows the error values and Figure 7 shows the computation times. The parameters used for the simulations are listed in Appendix A.

Figure 6 shows that the model selection filter with dynamic ρ had the lowest error, followed by model selection with static ρ, followed by the signal measure approach with dynamic ρ, followed by signal measure with static ρ. It may seem surprising that the error does not tend to decrease with time; however, the problem is high-observable which tends to result in almost immediate acquisition by the filter.

Figure 7 shows that the model selection filter is much faster than the signal measure. This is a result of the implementation, not the algorithm. If the filters were compared with equally optimized implementations, they should take approximately the same amount of computation time.

5. CONCLUSIONS

The results presented in section 4.2 allow us to make two important conclusions. First, unsurprisingly, the ρ optimization improves filter performance consistently. This demonstrates that the ρ graphs presented here, while they may not be completely optimal, are certainly better than using a static value.

More importantly, the results show that the model selection approach outperforms the signal measure approach. This is especially important because, previously, the signal measure approach has been the canonical choice for problems of this sort.

The flatness of the error (with respect to iteration) also reveals an important truth about high-observable problems like this one: the filter tends to have approximately the same error at the beginning of the simulation as at any other point. This suggests that it is not taking advantage of previous observations as well as it should. It is likely that this is because the particle weighting values can be so extreme in this sort of problem, that previous weights sometimes bear little importance in the filter calculations. It is likely that performance could be improved by considering this characteristic.

The computation time results provide little useful information except to show that the optimal ρ calculations slow the filter by a small but noticeable amount. This is due to the use of the neural net to store the optimal values. The values could be put into a lookup table, drastically reducing this overhead.

APPENDIX A. SIMULATION PARAMETERS

The first value listed for each parameter is the value used for the model selection filter, the second is the value used for the signal measure filter. If there is only one value, it was used for both filters.

A.1. Signal Parameters

The target speed, $S = 0.2$
The rate of turning for the target, $\lambda = 0.0005$
The amplitude of the target diffusion, $\sigma_s = 0.09$

A.2. Observation Parameters

The number of pixel cells, $n = 256$
The mean of the normal base of the noise, $\mu = 8.3$
The standard deviation of the normal base of the noise, $\sigma_o = 1.2$
The indicator for the presence of a target, $I_1 = 10000$
The indicator for the absence of a target, $I_0 = 2$

A.3. Filter Parameters

The number of particles, $N = 500000$
The static ρ used = 13000, 16000

A.4. Rho Optimization Parameters

Maximum $var = 4$, 10e18
Minimum $var = 0.0001$, 1e-66
Number of var discrete values = 10

Maximum $N_{\text{eff}} = 0.000002$
Minimum $N_{\text{eff}} = 0.00000000001$
Number of N_{eff} discrete values = 10

Maximum $var_t = 1000000000$
Minimum var_t = 1e-67
Number of var_t discrete values = 10

Maximum $\delta median$ value = 15000
Minimum $\delta median$ value = -20000
Number of $\delta median$ discrete values = 10

Maximum ρ value = 40000
Minimum ρ value = 5000
Number of ρ discrete values = 10

Data collection epochs = 3000
Iterations per epoch = 30

ACKNOWLEDGMENTS

The authors gratefully acknowledge the support and sponsorship of Lockheed Martin MS2 Tactical Systems and the Natural Science and Engineering Research Council (NSERC) through the Prediction in Interacting Systems (PINTS) centre of the Mathematics of Information Technology and Complex Systems (MITACS) network of centres of excellence.

REFERENCES

1. K. Yewchuk, C. Ketelson, A. Limon, Y. Mileyko, J. Hoffman, and M. Kouritzin. "Tracking and Identifying of Multiple Targets". PIMS graduate student industrial program workshop reports.
2. Y Zeng, and M. Kouritzin. "Bayesian Model Selection via Filtering for a Class of Micro-movement Models of Asset Price".
3. R. Kass and A. Raftery. "Bayes factors and model uncertainty". Journal of the American Statistical Association 90, 773-795. 1995.
4. D. Ballantyne, S. Kim, and M. Kouritzin, "A weighted interacting particle-based nonlinear filter", in Signal Processing, Sensor Fusion, and Target Recognition XI, ed. I. Kadar. Proceedings of SPIE 4729, 236-247. 2002.
5. W. Bauer, S. Kim, and M. Kouritzin, "Continuous and discrete space filters for predictions in accoustic positioning", in Proceedings of SPIE - image reconstruction from incomplete data II 4792, 193-206. 2002.
6. D.J. Ballantyne, J. Hailes, M.A. Kouritzin, H. Long, and J. Wiersma, "A hybrid weighted interacting particle filter for multi-target tracking", in Signal Processing, Sensor Fusion, and Target Recognition XII, 2003 Proceedings of SPIE 5096, 244-255.
7. T. Mitchell, "Machine Learning", McGraw Hill: Boston, Massachusetts, 1997.
8. J. S. Liu and R. Chen, "Sequential Monte Carlo methods for dynamical systems", J. Amer. Statist. Assoc., vol. 93, pp. 1032-1044, 1998.

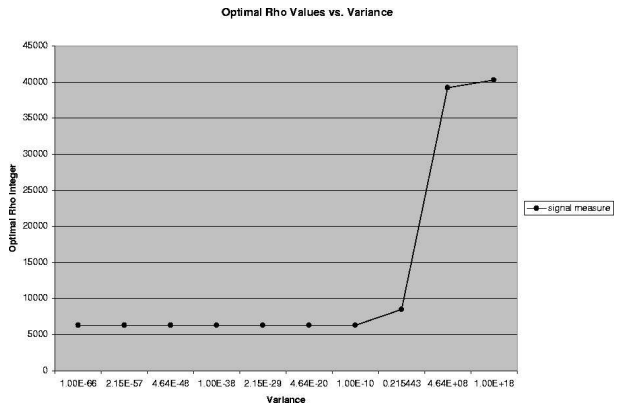

Figure 1. The optimal ρ values according to the neural network versus the variance value for the signal measure approach.

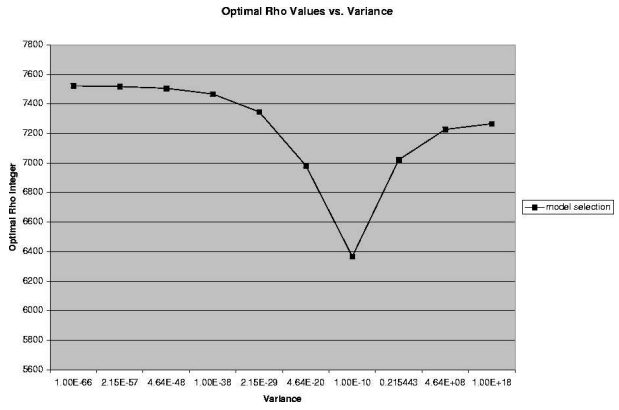

Figure 2. The optimal ρ values according to the neural network versus the variance value for the model selection approach.

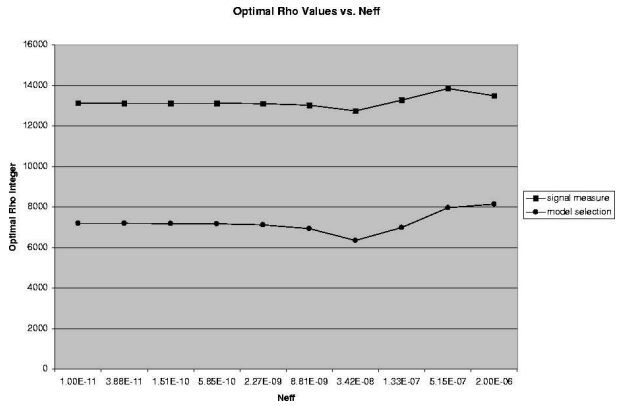

Figure 3. The optimal ρ values according to the neural network versus the N_{eff} value for both the signal measure and model selection approaches.

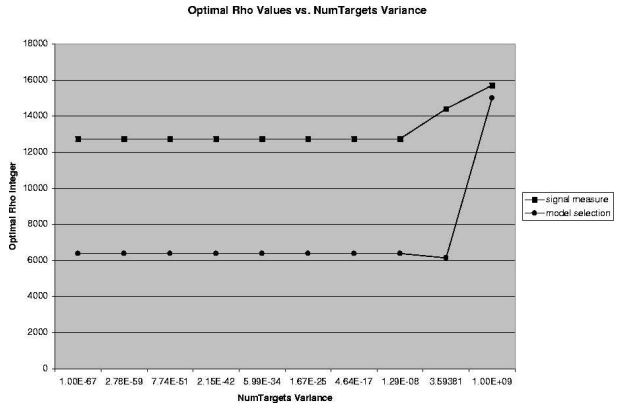

Figure 4. The optimal ρ values according to the neural network versus the number of targets' variance value for both the signal measure and model selection approaches.

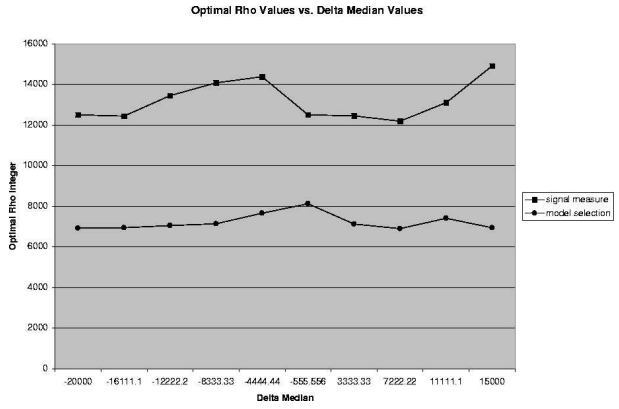

Figure 5. The optimal ρ values according to the neural network versus the $\delta median$ value for both the signal measure and model selection approaches.

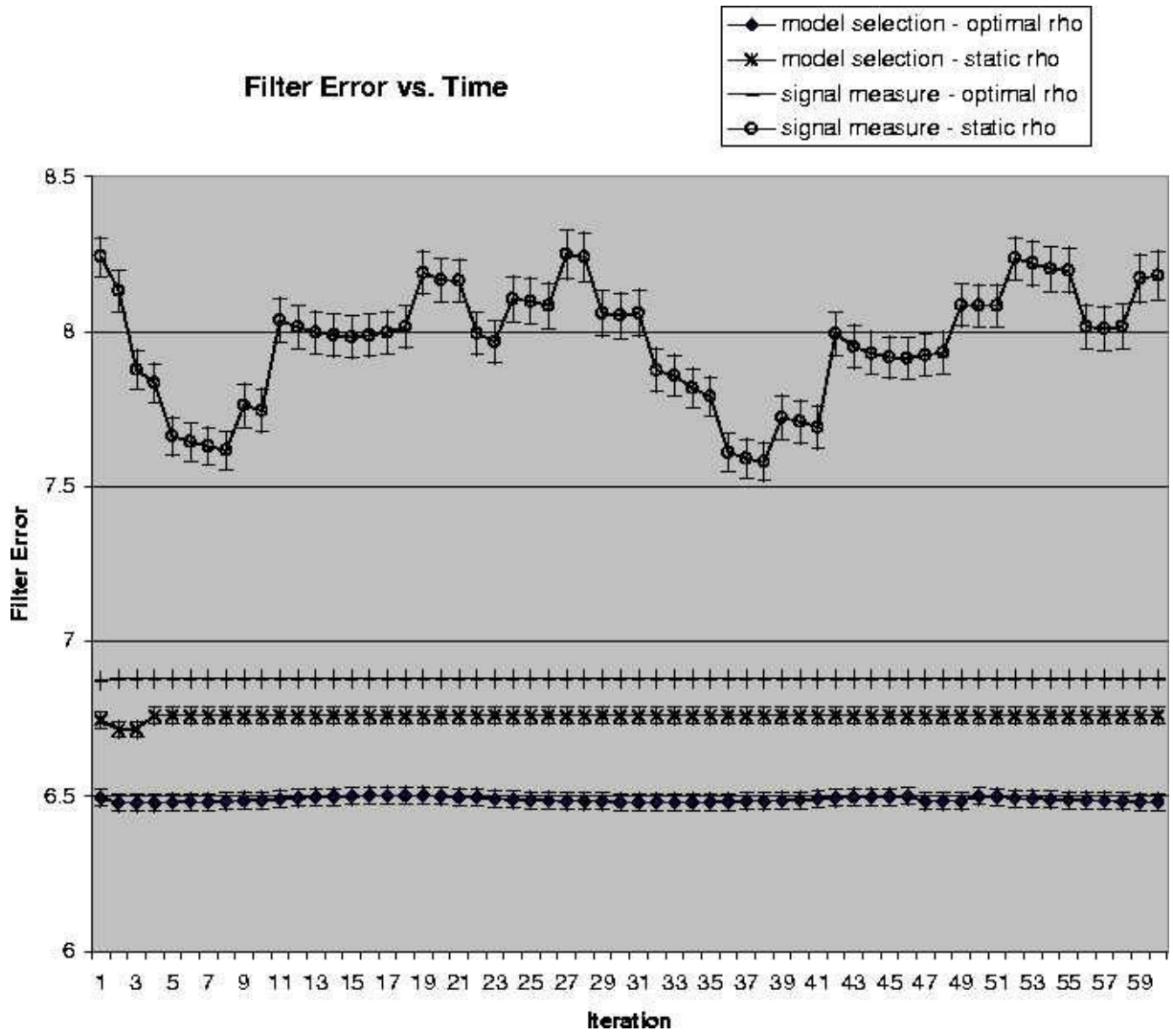

Figure 6. The filter error for each iteration. The error bars indicate a 95% confidence interval for the value.

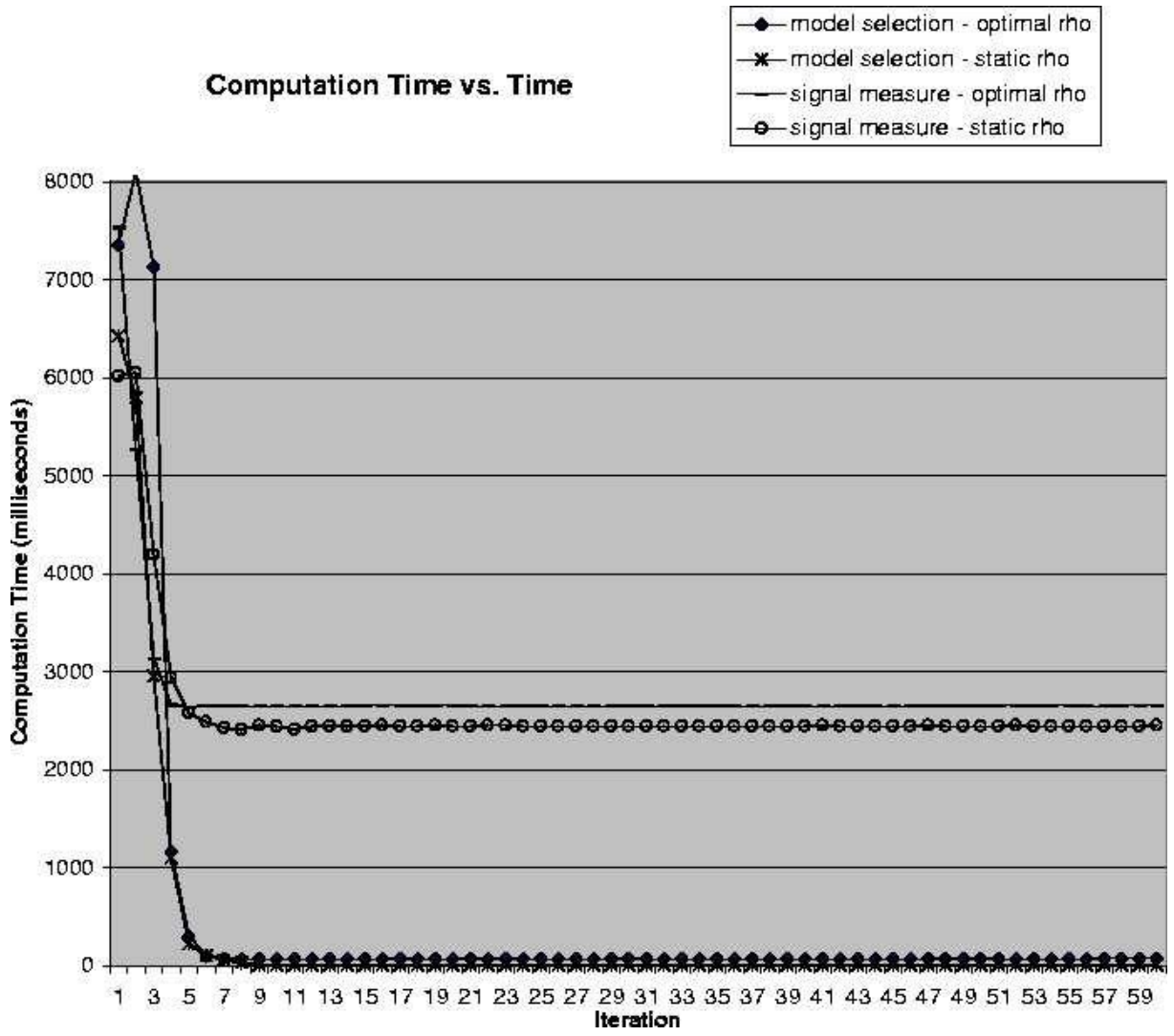

Figure 7. The computation time for each iteration. The error bars indicating a 95% confidence interval are too small to see.